



Réseaux nouveaux

Cours 2

Un cours de Yann Fornier

Use case : Le Magazine de Paul

Paris
34.78.9.15

Plan

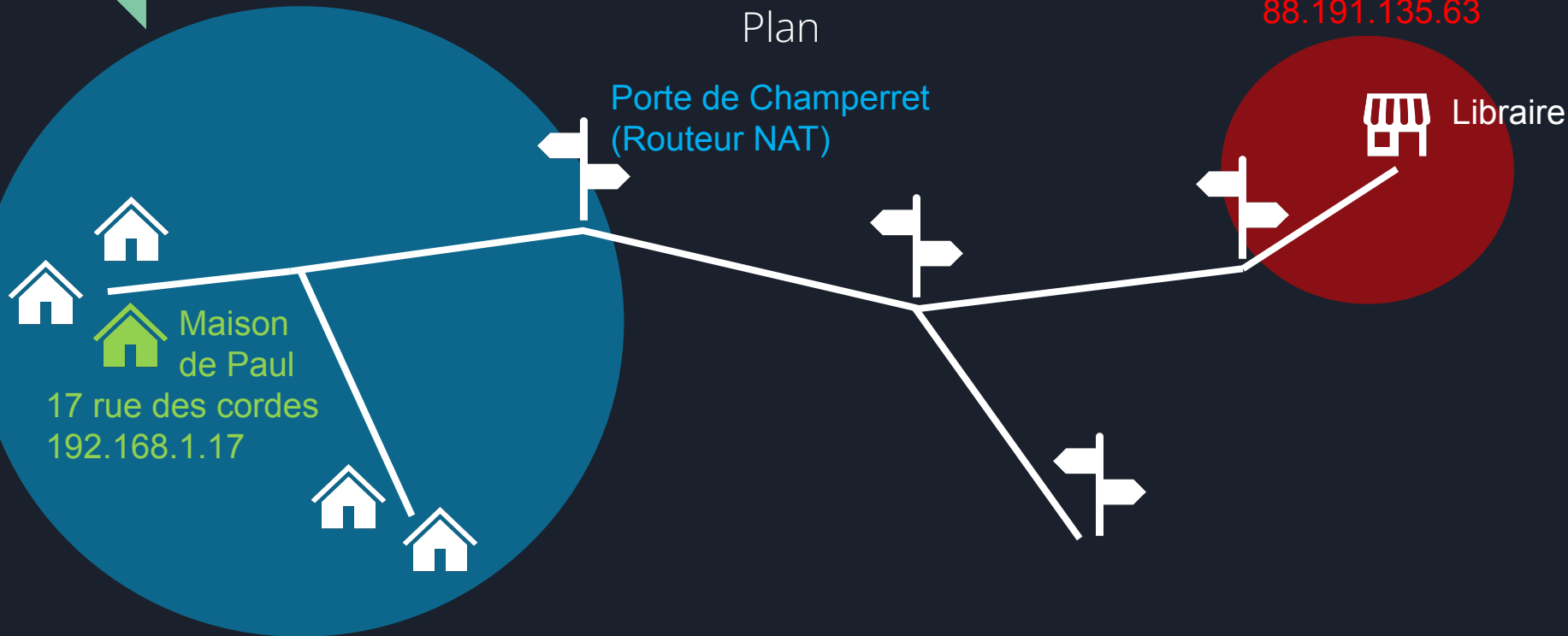
Nantes
88.191.135.63

Porte de Champerret
(Routeur NAT)

Libraire

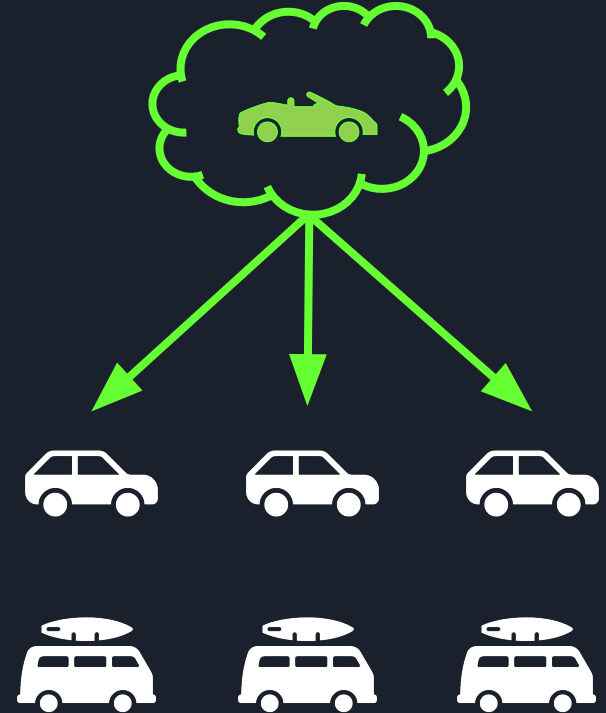
Maison
de Paul

17 rue des cordes
192.168.1.17



Cloud Manager

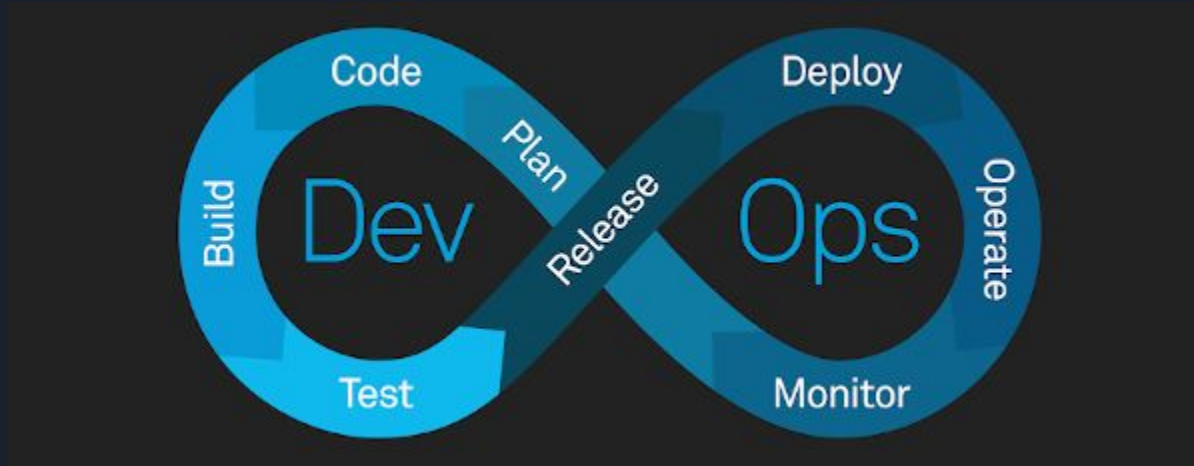
Le **Cloud Manager** est la centralisation des services de gestion de l'infrastructure accessible depuis le Cloud.



Fondements du DevOps



Définition et historique du DevOps



<https://devopssec.fr/article/introduction-au-devops#begin-article-section>



Définition et historique du DevOps

Le mouvement DevOps a émergé au début des années 2000 en réponse aux défis croissants posés par le développement logiciel et les opérations informatiques isolées. Les silos organisationnels traditionnels entre les équipes de développement et d'exploitation ont souvent conduit à des retards, des erreurs et des inefficacités dans le processus de livraison logicielle.

En 2003, Google embauche Ben Treynor (aujourd'hui Vice Président Ingénierie Google) en SRE (Site Reliability Engineering). Les SRE sont les premiers praticiens du DevOps d'aujourd'hui.



Définition et historique du DevOps

Combinant développement (Dev) et opérations (Ops), DevOps est l'union des personnes, des processus et des technologies destinés à fournir continuellement de la valeur aux clients.”

Microsoft

DevOps n'est ni un outil ni une technologie. Il s'agit plutôt d'une idéologie dans laquelle deux parties essentielles d'une entreprise : l'équipe de développement logiciel et l'équipe des opérations informatiques travaillent en étroite collaboration et partagent les progrès. DevOps garantit une bonne communication entre ces équipes, ce qui permet en outre à l'organisation de livrer le produit final en un minimum de temps et avec un minimum de problèmes.

Définition et historique du DevOps

L'idée de fusionner les aspects du développement et des opérations pour favoriser une approche plus collaborative a gagné en popularité grâce à des praticiens et des leaders de l'industrie. En 2009, Patrick Debois et Andrew Clay Shafer ont organisé la première conférence "DevOpsDays" à Ghent, en Belgique, marquant ainsi un jalon important dans la formalisation du mouvement DevOps.



Patrick Debois



Andrew Clay Shafer

Un schéma classique en entreprise



Applicative

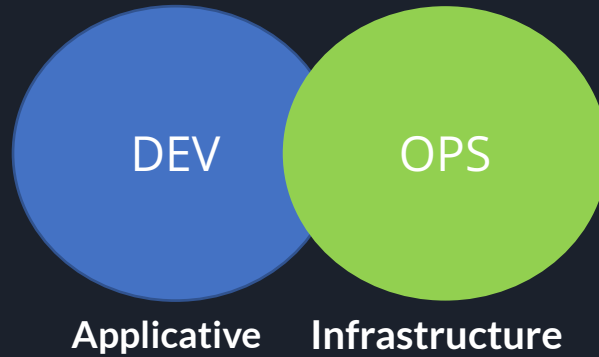


Infrastructure

Mur de la confusion




Le DevOps





Comment le DevOps aide les organisations ?

Le DevOps offre plusieurs avantages aux organisations en favorisant une approche collaborative entre les équipes de développement et d'exploitation, ainsi qu'en intégrant des pratiques et des outils visant à automatiser et optimiser le cycle de vie du développement logiciel.



Présentation des principaux concepts : collaboration, automatisation, intégration continue..

Livraison rapide et continue

Amélioration de la qualité

Collaboration renforcée

Automatisation des
processus

Réduction des coûts

Agilité et adaptabilité

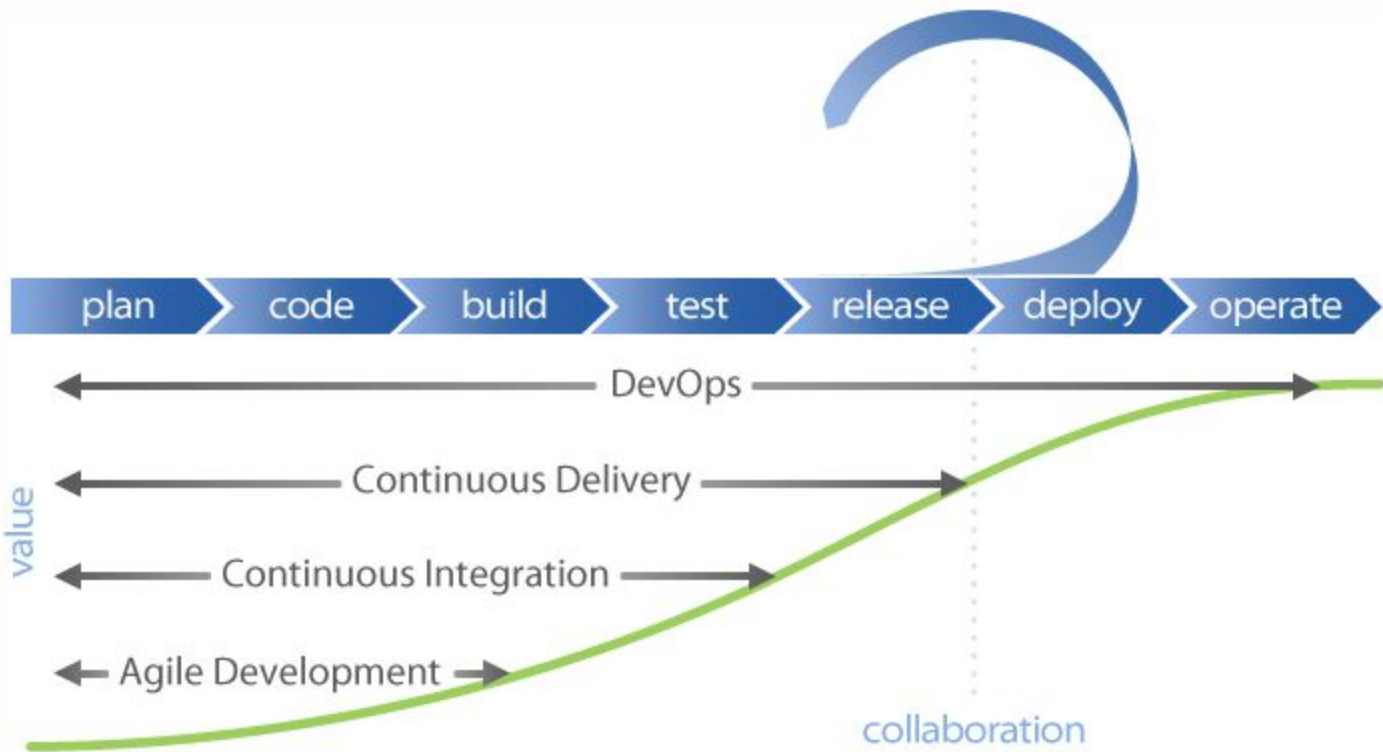
Surveillance et rétroaction
continues



Présentation des principaux concepts :

Livraison rapide et continue

Le DevOps vise à accélérer le processus de livraison logicielle en automatisant les tâches répétitives, en optimisant les workflows, et en intégrant des pratiques de déploiement continu. Cela permet aux organisations de fournir des fonctionnalités plus rapidement et de manière plus régulière.





Présentation des principaux concepts :

L'amélioration de la qualité

L'intégration continue, les tests automatisés et la surveillance constante contribuent à une meilleure qualité du code et à la détection précoce des erreurs. Cela réduit les risques d'incidents en production et améliore la stabilité des applications.



Présentation des principaux concepts :

La collaboration renforcée

Le DevOps encourage la collaboration étroite entre les équipes de développement, d'exploitation et d'autres parties prenantes. En favorisant la communication et le partage des responsabilités, le DevOps brise les silos organisationnels et contribue à une culture de travail plus collaborative.



Présentation des principaux concepts :

L'automatisation des
processus

L'automatisation des tâches manuelles, telles que le déploiement, les tests, et la gestion de l'infrastructure, permet de gagner du temps et de réduire les erreurs humaines. L'automatisation est au cœur du DevOps pour assurer une efficacité opérationnelle accrue.



Présentation des principaux concepts :

La réduction des coûts

En accélérant le cycle de vie du développement logiciel et en automatisant les processus, le DevOps contribue à une utilisation plus efficiente des ressources, ce qui peut se traduire par une réduction des coûts opérationnels.



Présentation des principaux concepts :

L'agilité et l'adaptabilité

Le DevOps permet aux organisations de s'adapter plus rapidement aux changements du marché et aux exigences des clients. La flexibilité accrue dans le développement et le déploiement facilite l'ajustement rapide des fonctionnalités en réponse aux besoins évolutifs.



Présentation des principaux concepts :

La Surveillance et la
rétroaction continue

Les pratiques DevOps incluent la surveillance constante des performances et des opérations en production, ainsi que la collecte continue de données et de retours d'expérience. Cela permet aux équipes d'identifier rapidement les problèmes, de les résoudre et de faire de l'itération en continu pour améliorer la performance et la satisfaction client.

Les avantages du DevOps





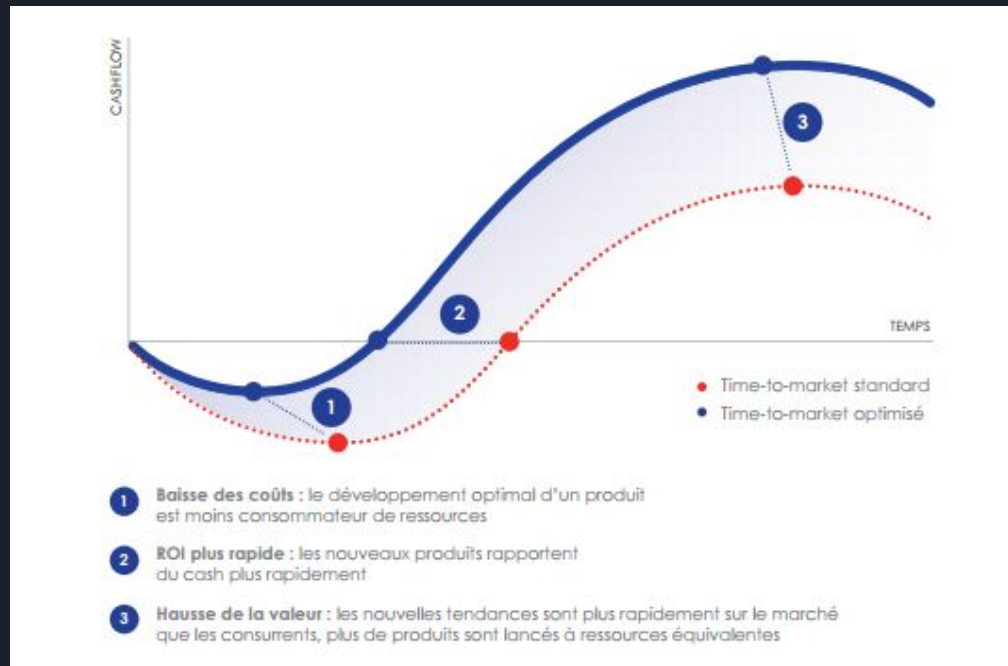
La collaboration

Les principales problématiques des entreprises étaient sur le manque de collaboration entre les équipes de Dev et les équipes Opérationnelles (Ops). L'avantage du DevOps réside dans la collaboration entre ces deux équipes en repensant intégralement l'environnement où ces différentes équipes travaillent afin de créer davantage de valeur pour l'entreprise.

La vitesse

Le DevOps d'aujourd'hui permet d'accélérer la fréquence et la vitesse à laquelle les entreprises peuvent introduire de nouveaux produits afin d'obtenir un avantage concurrentiel.

Cette réduction de temps est liée à ce que l'on appelle le TTM (Time-To-Market)





L'Agilité pour la satisfaction client

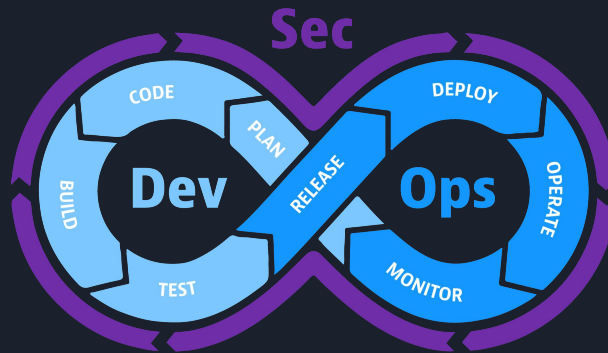
Les exigences des clients modernes sont intenses et demandent une très forte réactivité quant au développement de nouvelles fonctionnalités. Les pratiques DevOps permettent une meilleure flexibilité et aide également les équipes à comprendre comment les clients se servent de leurs produits.

En général les clients détestent attendre les produits d'une entreprise. Plus il y a d'attente plus les effets néfastes s'intensifient. Avec un rythme de livraison plus rapide, la satisfaction client augmente plus rapidement.

La sécurité

Le DevOps permet dans sa version optimisée d'intégrer la sécurité des produits livrés. On parle alors de DevSecOps.

Le DevSecOps étend les principaux composants de développement et d'exploitation du DevOps et y introduit une couche de sécurité en tant que composant distinct dans le pipeline CI/CD. l'essence du DevSecOps est que la majorité des équipes (pas seulement les équipes de sécurité) sont responsables de la sécurité de l'application. Il aide à réduire les coûts et avec lui les équipes sont en mesure de suivre et de détecter les problèmes de sécurité dans les premiers stades de développement.





Ce que n'est pas le DevOps

Une équipe distincte

Un Outil

Une simple combinaison d'
équipes Dev et Ops

Une stratégie universelle

Uniquement
l'automatisation



Conclusion

Le DevOps permet d'améliorer la collaboration entre toutes les parties prenantes de la planification à la livraison et l'automatisation du processus de livraison afin de :

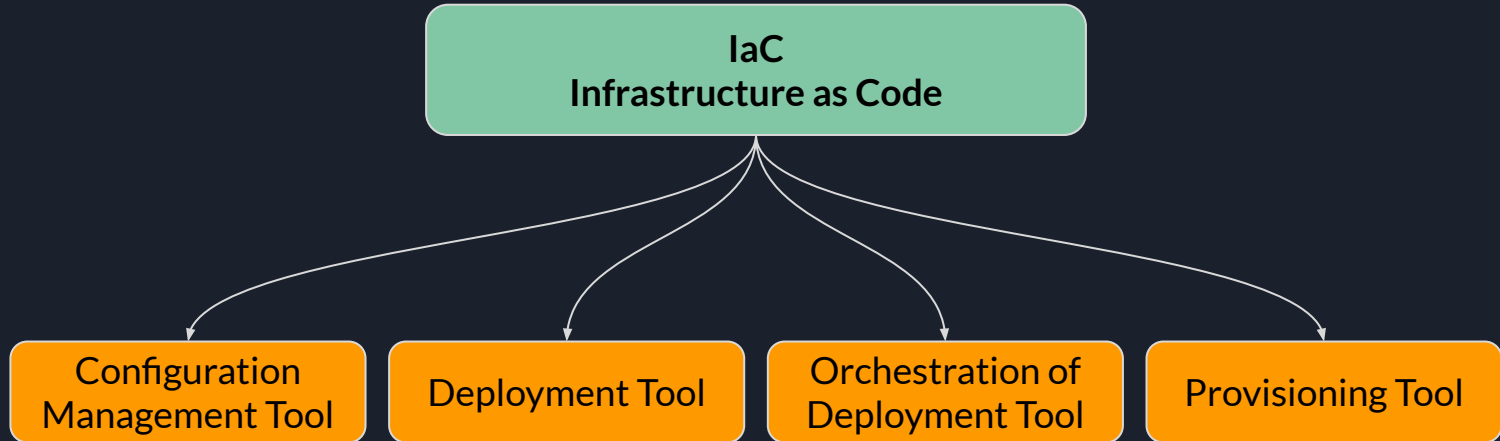
- Améliorer la fréquence de déploiement
- Accélérer la mise sur le marché
- Réduire le taux d'échec des nouvelles livraisons
- Raccourcir le délai entre les correctifs
- Améliorer le temps moyen de récupération
- S'adapter aux changements des besoins client avec l'agilité
- Posséder un avantage concurrentiel
- Satisfaire les clients
- Accroître l'innovation
- Améliorer la sécurité

L'Infrastructure as Code (IaC)



Concepts fondamentaux d'IaC

Une des mises en pratiques du DevOps





L'automatisation : Quelques définitions

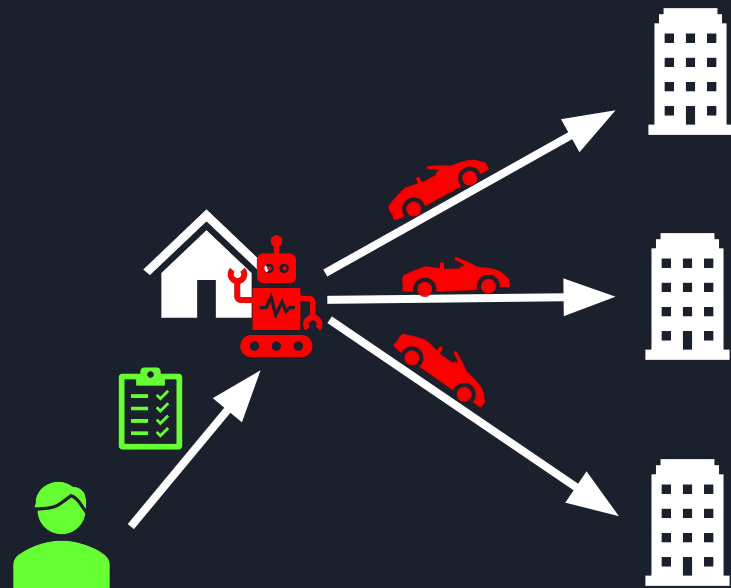
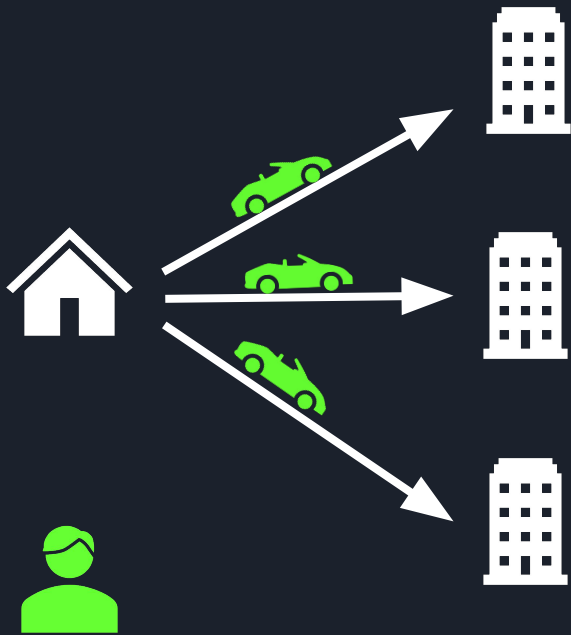
Automatisation : Ensemble des procédés automatiques visant à supprimer l'intervention humaine dans la production industrielle et le traitement de l'information.

Automatique: Se dit d'un dispositif qui exécute de lui-même certaines opérations définies à l'avance.

Automate: Appareil équipé de dispositifs qui permettent l'exécution de certaines tâches sans intervention humaine.

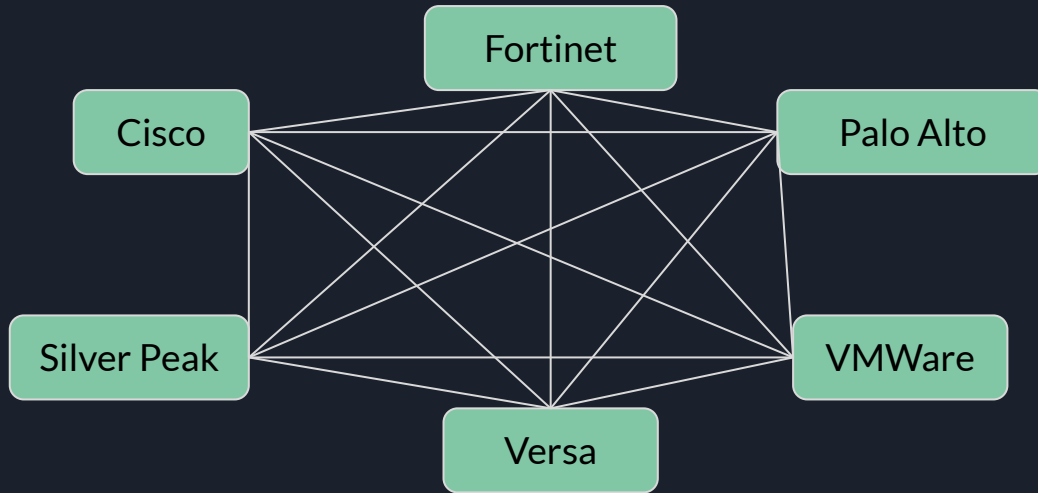


Automate



Pourquoi l'infrastructure en a besoin ?

Souvent hétérogène, plus une infrastructure est importante, plus il est compliqué de la maintenir sans problèmes. Elle demande des connaissances spécifiques sur chaque équipement.



SD-WAN

	Cisco	VMware	Fortinet	Silver Peak	Versa	Palo Alto Networks (Prisma SD-WAN)	Palo Alto Networks (PA-OS NGFW)
Tout développer							
^ Réseau							
+ Prise en charge du routage classique et du SD-WAN sur la même plateforme	✓	✗	✓	✗	✗	✗	✓
+ SD-WAN au cœur du réseau, à la périphérie et dans le cloud	✓	✓	✓	✓	✓	✓	✓
+ Une architecture SD-WAN personnalisée	✓	✗	✗	✗	✓	✗	✗
+ Un provisionnement réellement automatique	✓	⚠	✗	⚠	✗	✓	⚠
+ Topologie SD-WAN à routeur double actif-actif	✓	✗	⚠	⚠	✗	✗	✓
+ Protocoles de routage avancé pour les intégrations brownfield	✓	⚠	✓	⚠	✓	⚠	✓

Philosophie Pets vs Cattle (Randy Bias)

Pets

Serveurs que l'on traite de manière unique
et qui ne peuvent jamais tomber.
Modèle hétérogène



Cattle

Serveurs déployés par des outils
d'automatisation. Réplica des serveurs. Ils
peuvent être remplacés à l'identique
Modèle homogène



Les Outils

PERIODIC TABLE OF DEVOPS TOOLS (V3)

1 Os GI GitLab																			2 En Sp Splunk	
3 Fm Gh GitHub	4 En Dt Datical																			10 Fm Sg Sumo Logic
11 Os Sv Subversion	12 En Db DBMaestro																			18 Os Fd Fuentd
19 En Cw ISPW	20 En Dp Delphix	21 Os Jn Jenkins	22 Fm Cs Codeship	23 Os Fn FitNesse	24 Fr Ju JUnit	25 Fr Ka Karma	26 Fm Su SoapUI	27 En Ch Chef	28 Fr Tf Terraform	29 En XLd XebiaLabs XL Deploy	30 En Ud UrbanCode Deploy	31 Os Ku Kubernetes	32 Fm Cc CA CD Director	33 En Pr Pitrua Release	34 Pd Al Alibaba Cloud	35 Fm Os OpenStack	36 Os Ps Prometheus			
37 Pd At Artifactory	38 Fm Rg Redgate	39 Pd Ba Bamboo	40 Fm Vs VSTS	41 Fr Se Selenium	42 Fr Jm JMeter	43 Os Ja Jasmine	44 Pd Sl Sauce Labs	45 En An Ansible	46 Os Ru Rudder	47 En Oc Octopus Deploy	48 Os Go GoCD	49 Os Ms Mesos	50 Pd Gke GKE	51 Pd Om OpenMake	52 Pd Cp AWS CodePipeline	53 Pd Cy Cloud Foundry	54 En It ITRS			
55 Pd Nx Nexus	56 Os Fw Flyway	57 Os Tr Travis CI	58 Fm Tc TeamCity	59 Os Ga Gatling	60 Fr Tn TestNG	61 Fm Tt Tricentis Tosca	62 Pd Pe Perfecto	63 En Pu Puppet	64 Os Pa Packer	65 Fm Cd AWS CodeDeploy	66 En Ec ElectricCloud	67 Os Ra Rancher	68 Pd Aks AKS	69 Os Rk Rkt	70 Os Sp Spinnaker	71 Os Ir Iron.io	72 Pd Mg Mooqsoft			
73 Fm Bb BitBucket	74 En Pf Perforce	75 Fm Cr CircleCI	76 Pd Cb AWS CodeBuild	77 Fr Cu Cucumber	78 Os Mc Mocha	79 Os Lo Locust.io	80 En Mf Micro Focus UFT	81 Os Sa Salt	82 Os Ce CFEngine	83 En Eb ElasticBox	84 En Ca CA Automic	85 En De Docker Enterprise	86 Pd Ae AWS ECS	87 Fm Cf Codefresh	88 Os Hm Helm	89 Os Aw Apache OpenWhisk	90 Os Ls Logstash			

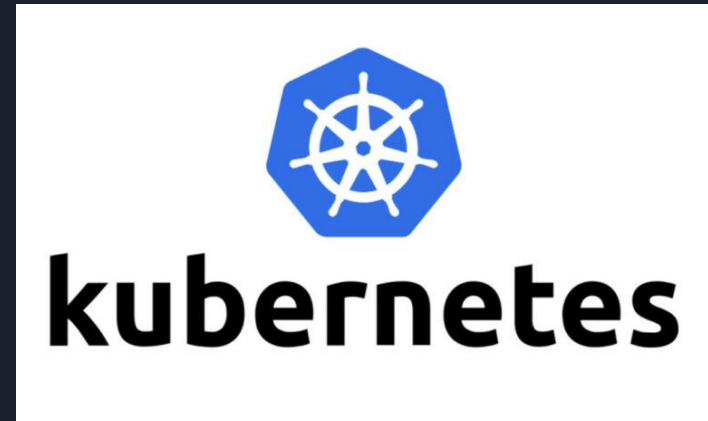
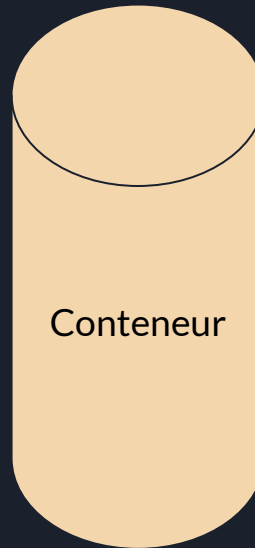


 Follow @xebialabs

91	En	92	Os	93	Fm	94	En	95	En	96	Fm	97	Os	98	Os	99	Os	100	En	101	En	102	En	103	En	104	Os	105	Os
XLi		Ki		Nr		Dt		Dd		Ad		El		Ni		Zb		Zn		Cx		Sg		Bd		Sr		Hv	
XebiaLabs XL Impact		Kibana		New Relic		Dynatrace		Datadog		AppDynamics		ElasticSearch		Nagios		Zabbix		Zenoss		Checkmarx SAST		Signal Sciences		BlackDuck		SonarQube		HashiCorp Vault	
106	En	107	Pd	108	Fm	109	Fm	110	Fm	111	En	112	En	113	En	114	Pd	115	Pd	116	En	117	Fm	118	En	119	En	120	En
Sw		Jr		Tl		Sk		St		Cn		Ry		Ac		Og		Pd		Sn		Tw		Ck		Vc		Ff	
ServiceNow		Jira		Trello		Slack		Stride		CollabNet VersionOne		Remedy		Agile Central		OpsGenie		Pagerduty		Snort		Tripwire		CyberArk		Veracode		Fortify SCA	

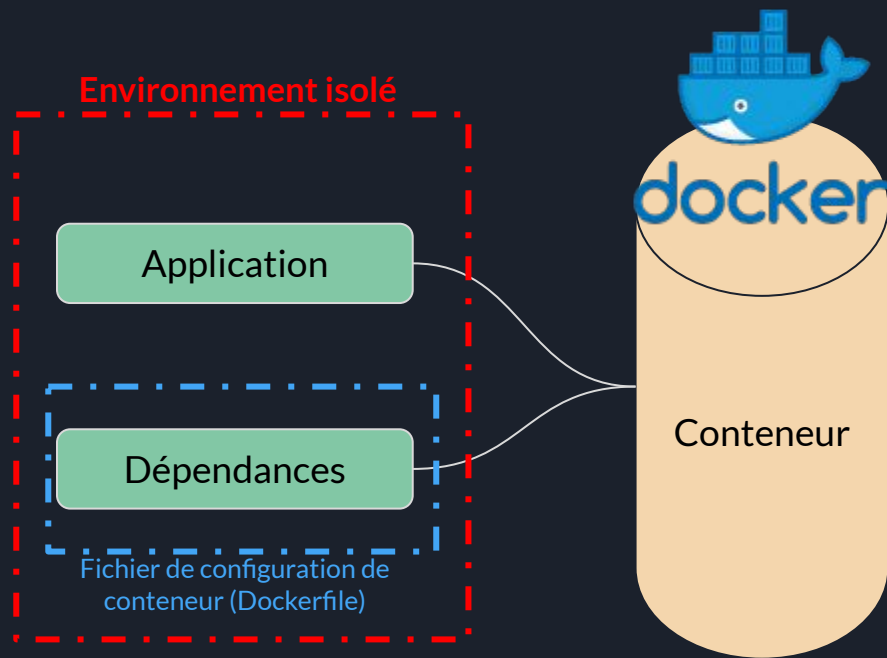


Qu'est ce qu'un conteneur ?



Introduction à la conteneurisation

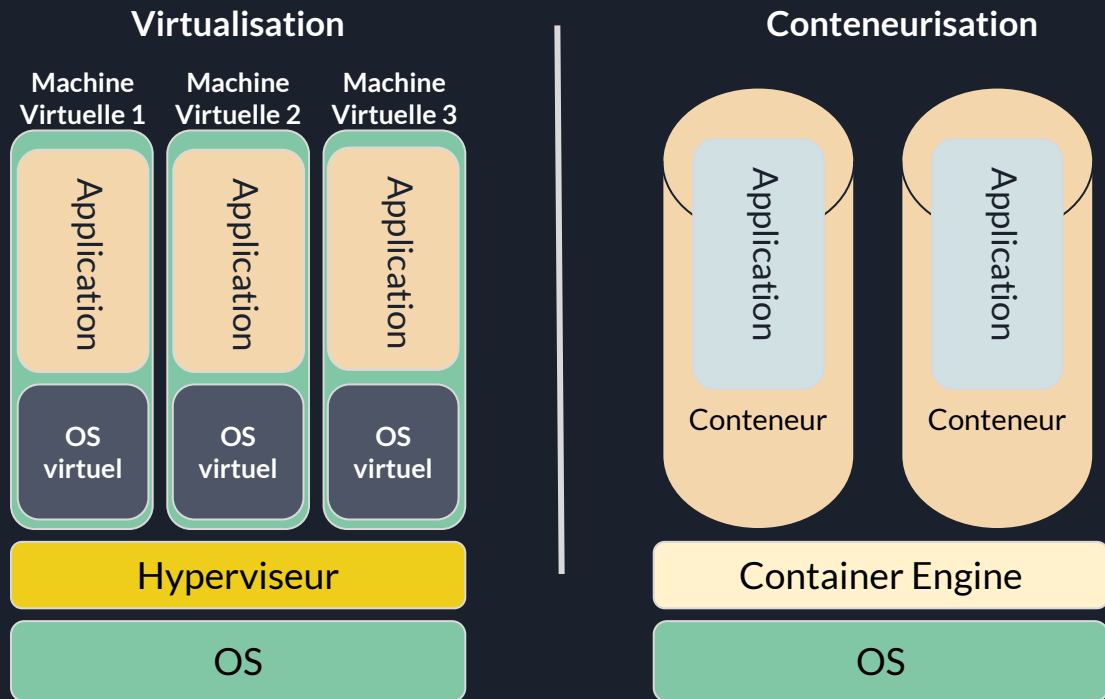
Qu'est ce qu'un conteneur ?



Un conteneur est un environnement logiciel qui permet de lancer et exécuter des applications de manière isolée et autonome. Il encapsule les applications, leurs dépendances et leur configuration dans un environnement qui peut être facilement déplacé d'un système à un autre.

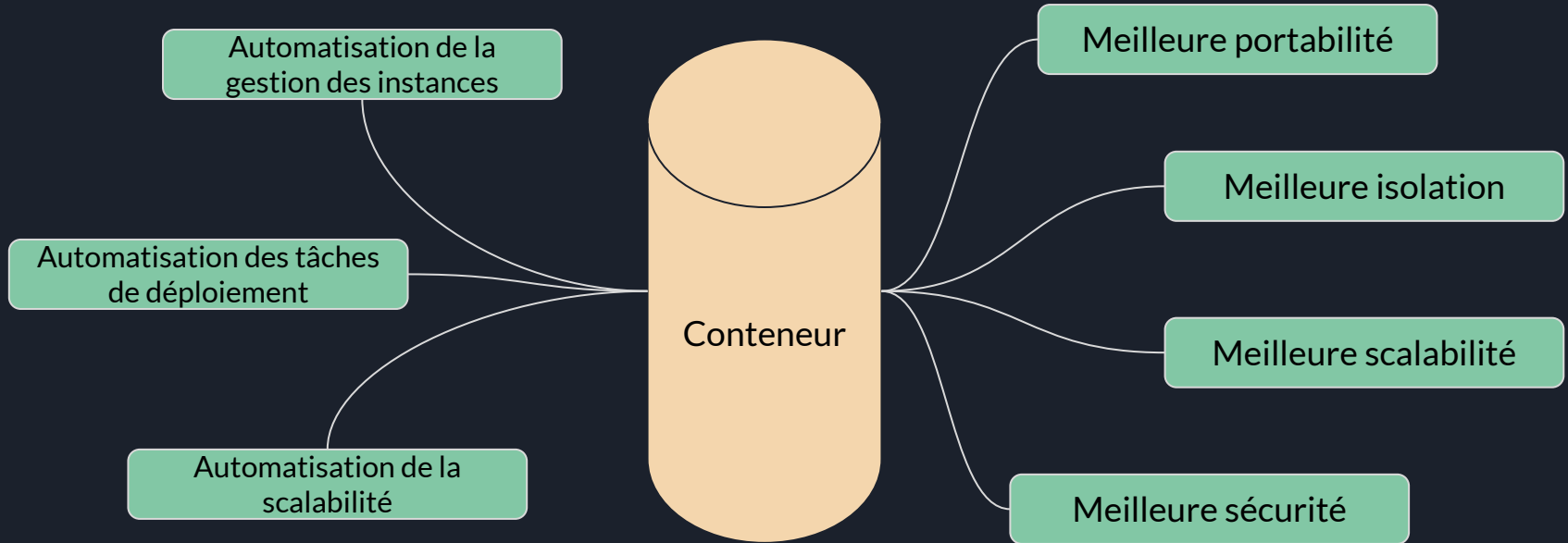
Introduction à la conteneurisation

Qu'est ce qu'un conteneur ?

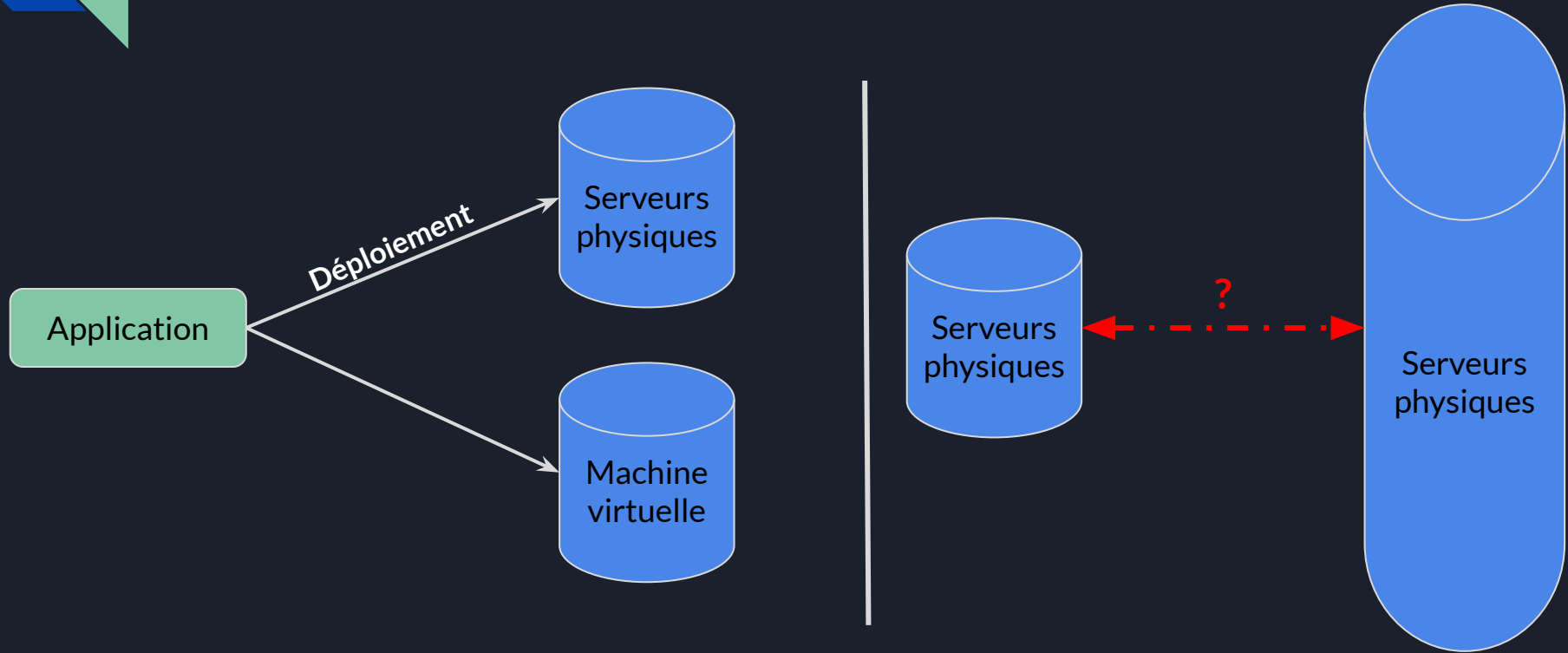


Les conteneurs sont similaires aux machines virtuelles, mais contrairement à ces dernières, ils ne nécessitent pas de système d'exploitation virtuel, ils partagent donc le système d'exploitation hôte, ce qui les rend plus légers et plus rapides à démarrer.

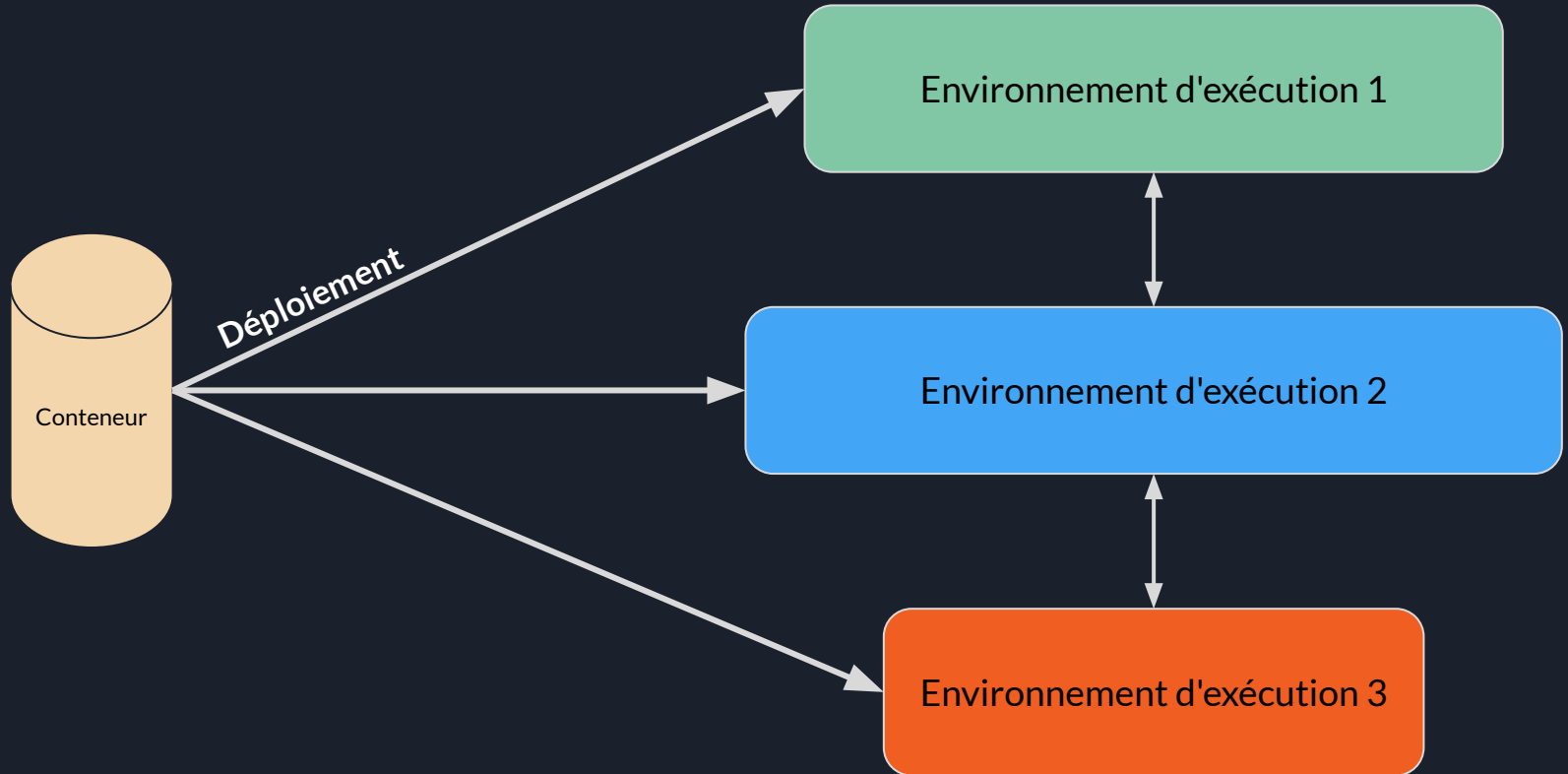
Qu'est ce qu'un conteneur ?



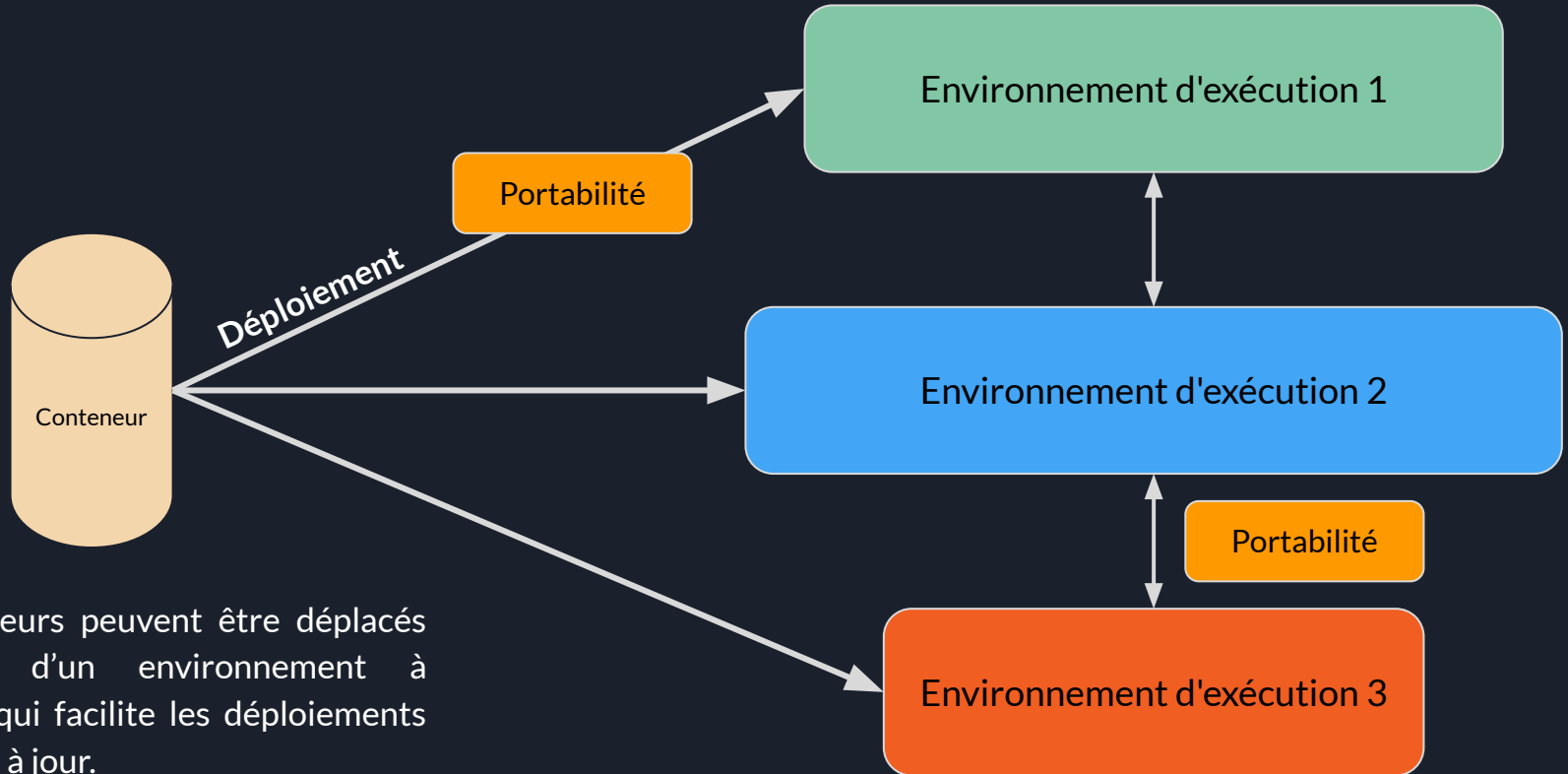
Les défis de l'environnement d'exécution des applications



Les avantages de la conteneurisation

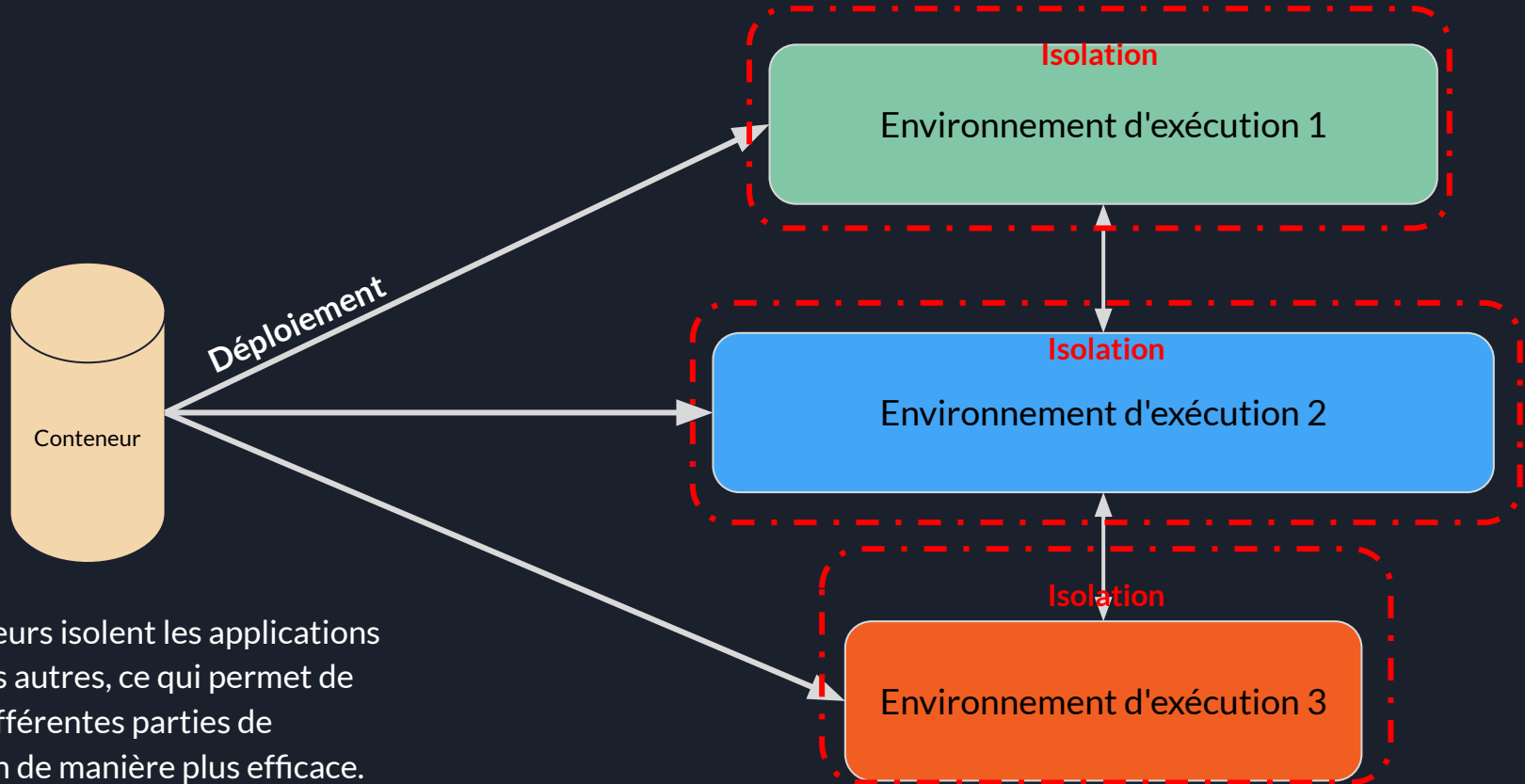


Les avantages de la conteneurisation



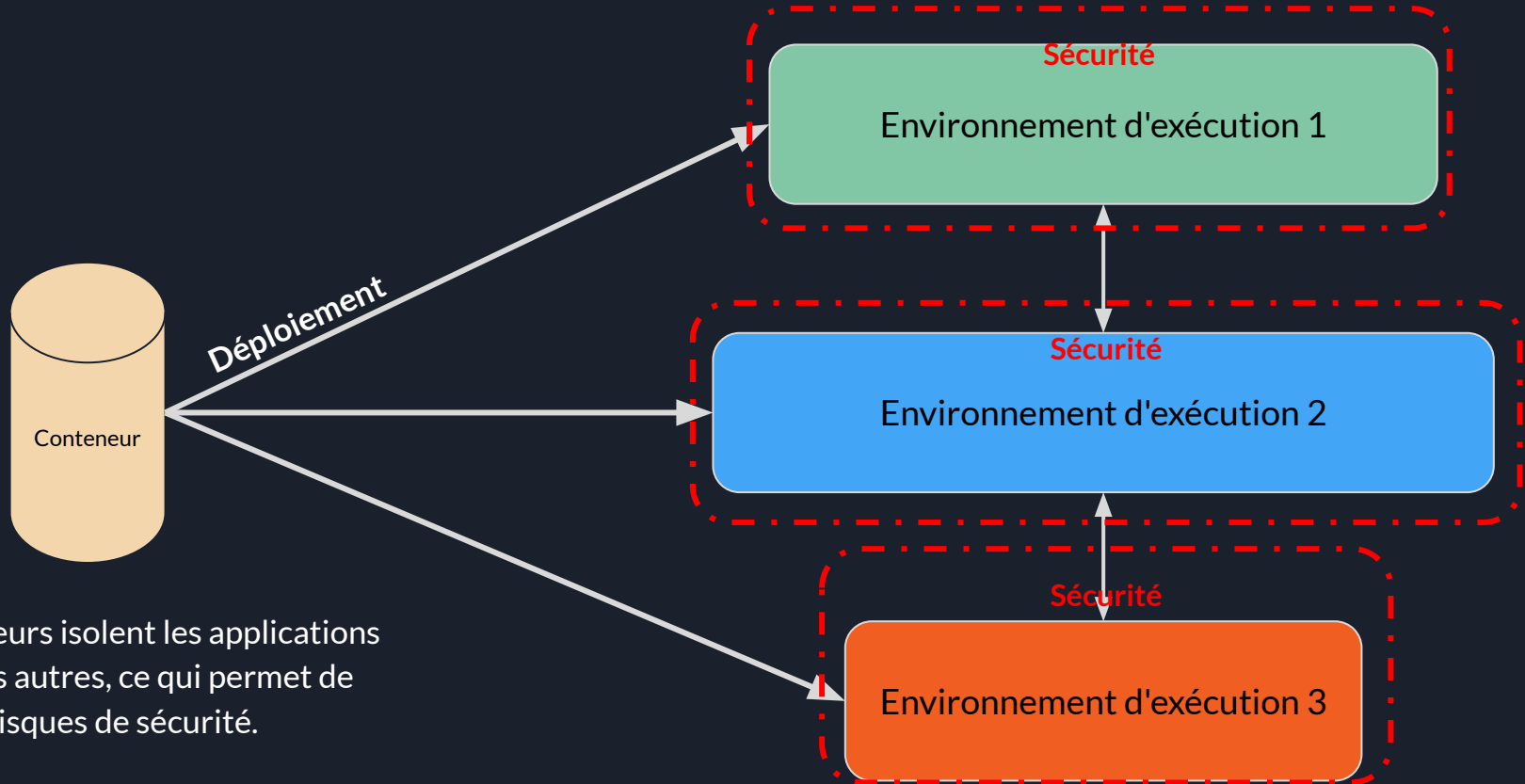
Les conteneurs peuvent être déplacés facilement d'un environnement à l'autre, ce qui facilite les déploiements et les mises à jour.

Les avantages de la conteneurisation



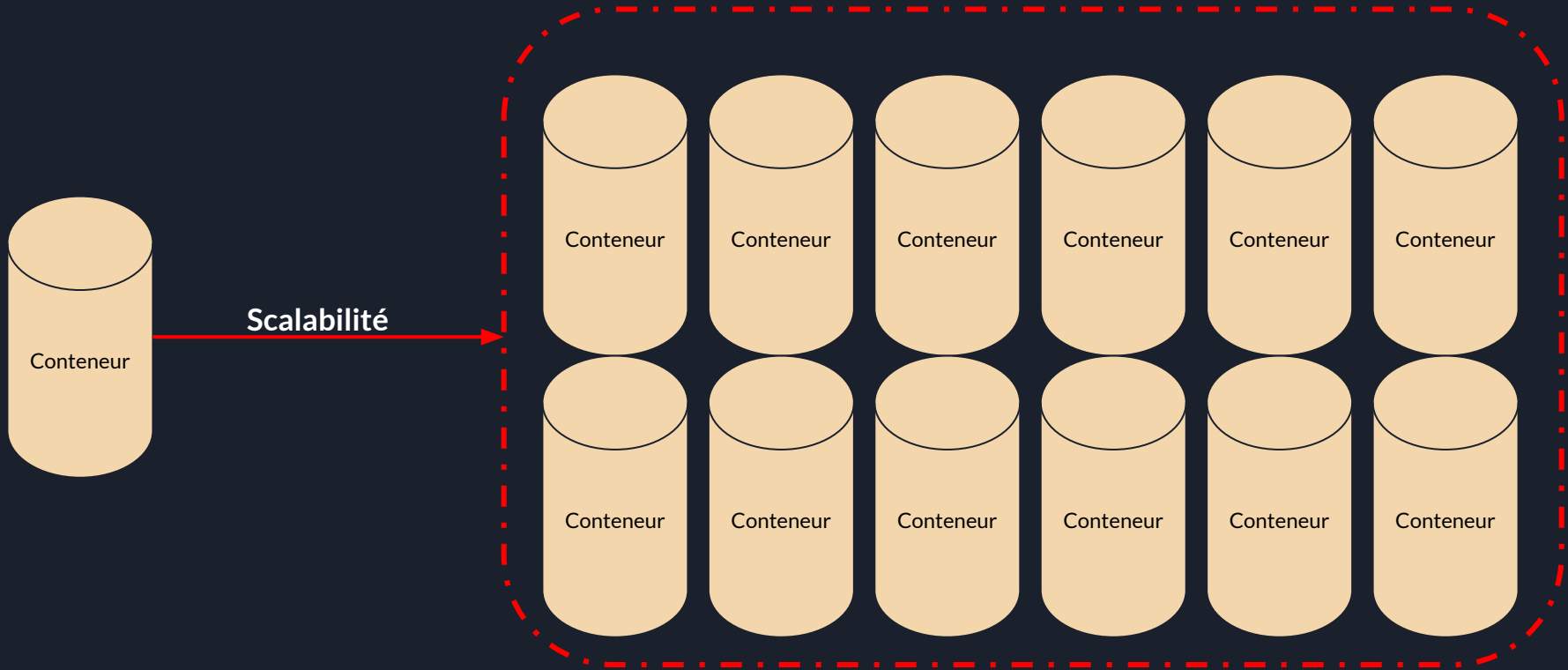
Les conteneurs isolent les applications les uns des autres, ce qui permet de gérer les différentes parties de l'application de manière plus efficace.

Les avantages de la conteneurisation



Les conteneurs isolent les applications les unes des autres, ce qui permet de limiter les risques de sécurité.

Les avantages de la conteneurisation






Conclusion

En résumé, la conteneurisation est une technique de “virtualisation légère” qui permet d’isoler les applications et leurs dépendances dans des environnements autonomes appelés conteneurs.

Cela permet de garantir que l’application fonctionne de manière identique sur tous les environnements, qu’il s’agisse d’un environnement de développement, de test ou de production. Il offre des avantages tels que la portabilité, l’isolation, la scalabilité et la sécurité.

Il est souvent comparé aux machines virtuelles qui ont besoin de plus de ressources et peuvent être plus lents à démarrer, tandis que les conteneurs sont plus légers et plus rapides, ils sont donc plus adaptés pour les applications qui nécessitent une scalabilité et une flexibilité élevées.



Les concepts clés de la conteneurisation : images, conteneurs, registres

La conteneurisation repose sur trois concepts clés : les images, les conteneurs et les registres.



Images



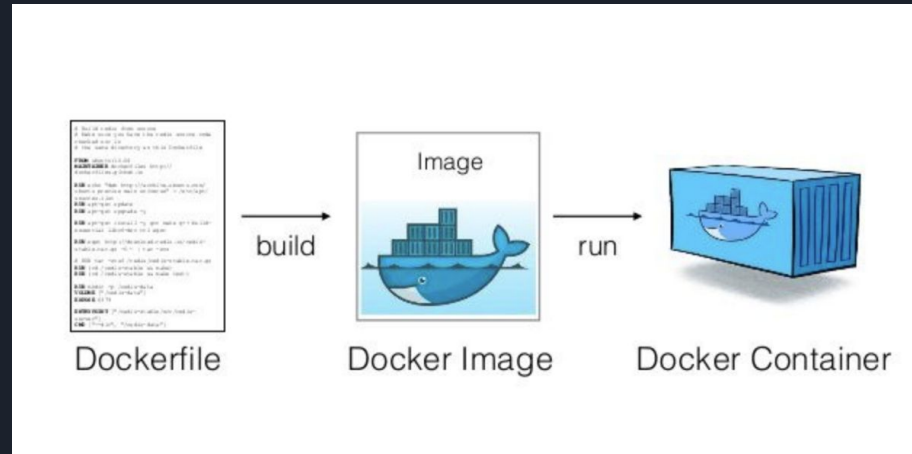
Conteneurs



Registres

Les concepts clés de la conteneurisation : images, conteneurs, registres

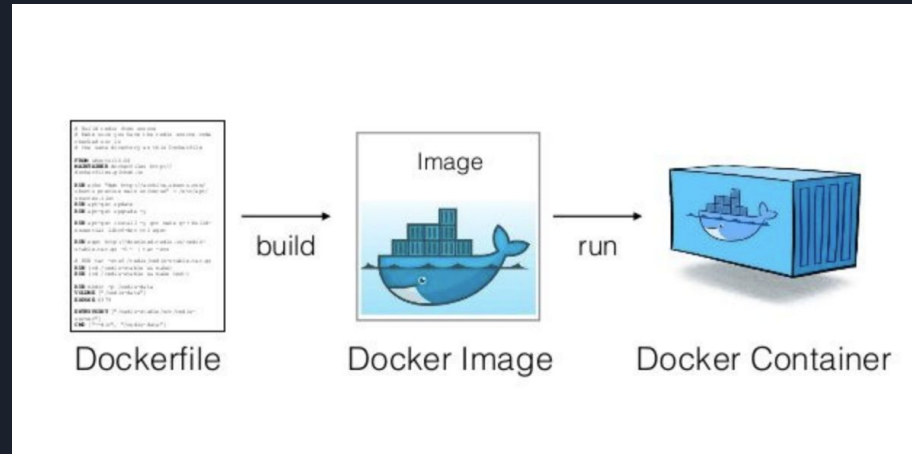
Une image est un package qui contient une application et toutes ses dépendances, ainsi que les instructions nécessaires pour créer un conteneur à partir de l'image.



Les concepts clés de la conteneurisation : images, conteneurs, registres

Un conteneur est une instance en cours d'exécution d'une image. Les conteneurs sont légers et portables, et peuvent être déployés rapidement et facilement.

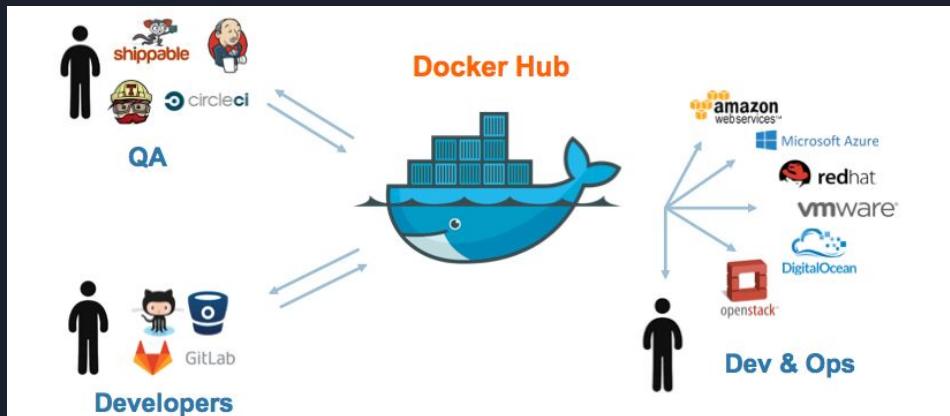
Conteneurs



Les concepts clés de la conteneurisation : images, conteneurs, registres

Un registre est un dépôt centralisé pour stocker et distribuer des images Docker. Le registre public de Docker, Docker Hub, contient des milliers d'images pré-construites que les développeurs peuvent utiliser pour créer des conteneurs.

Registres



Access the world's largest library of container images



busybox
Official
↓ 1B+



ubuntu
Official
↓ 1B+



python
Official
↓ 1B+



postgres
Official
↓ 1B+



memcached
Official
↓ 1B+



httpd
Official
↓ 1B+



mysql
Official
↓ 1B+



traefik
Official
↓ 1B+



rabbitmq
Official
↓ 1B+



docker
Official
↓ 1B+

[See all Docker Official Images](https://hub.docker.com/)

<https://hub.docker.com/>

Introduction à Docker

Docker est la technologie de conteneurisation la plus populaire et la plus largement utilisée.



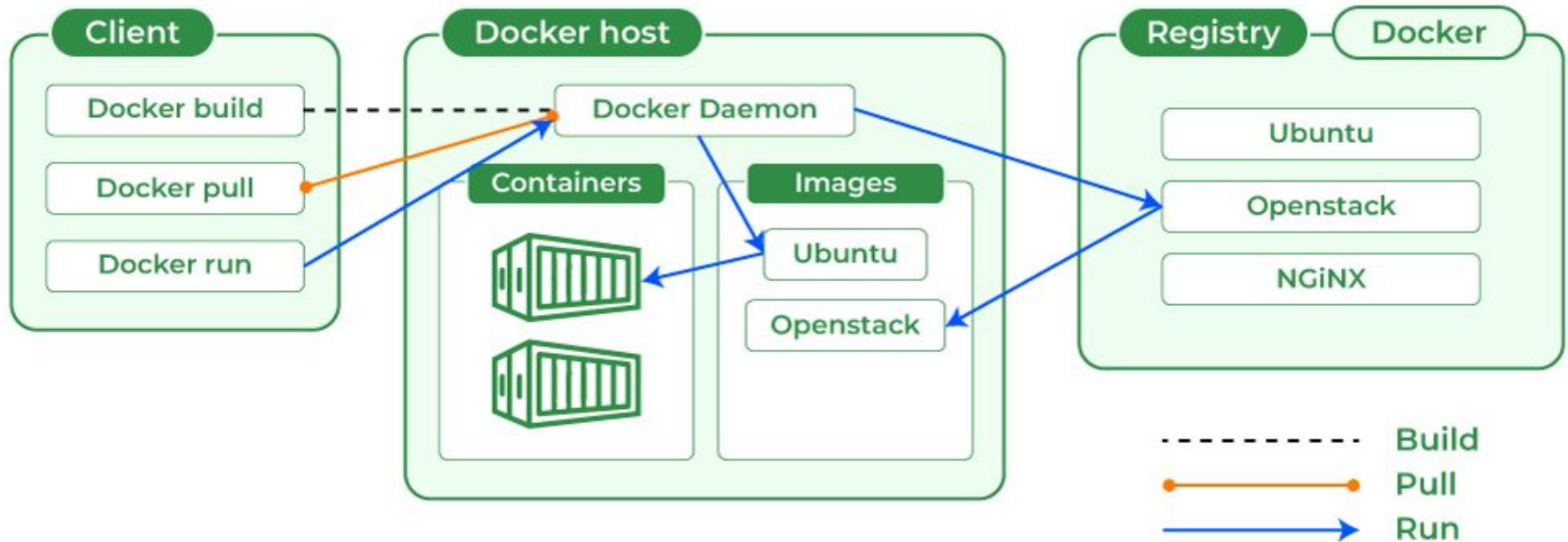
Histoire de Docker

Docker a été créé en 2013 par Solomon Hykes et est rapidement devenu un outil incontournable dans l'écosystème des applications web. La popularité de Docker est due en partie à sa facilité d'utilisation et à sa portabilité.



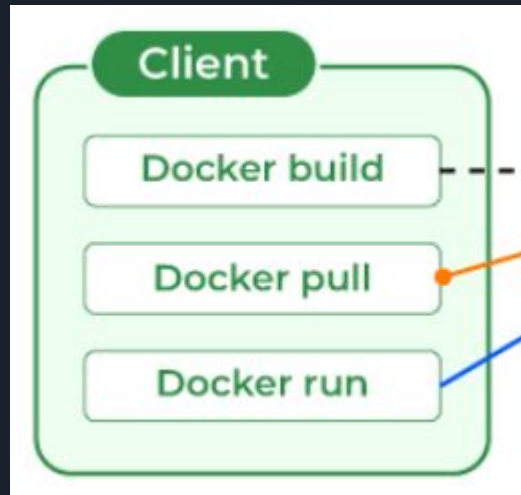
Solomon Hykes

Les composants de Docker



Le client Docker

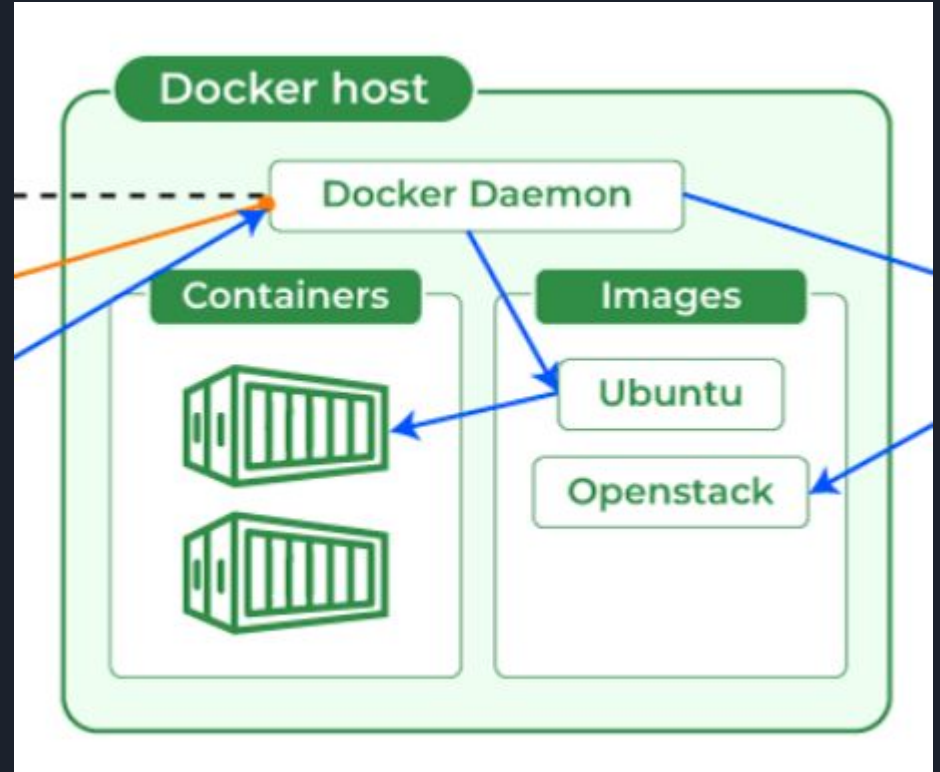
Le client Docker est la principale interface permettant de communiquer avec le système Docker. il reçoit les commandes par la CLI et les transmet au Docker Daemon.



Le Docker host

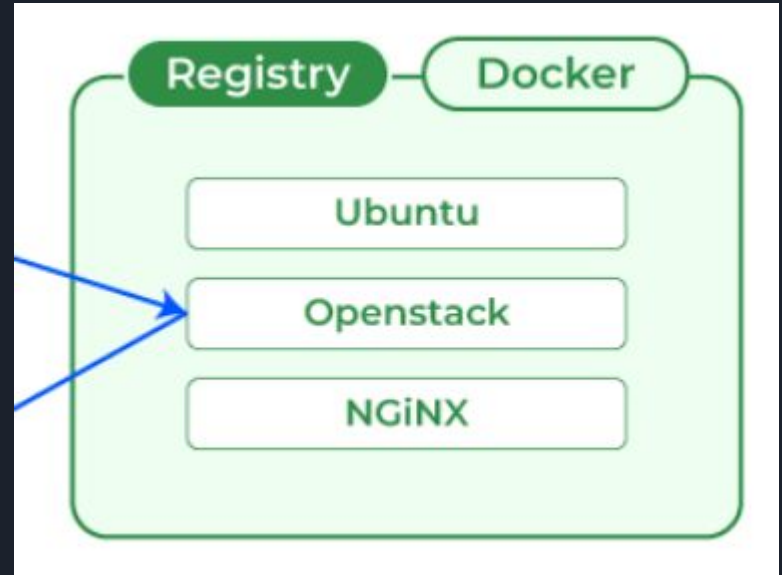
Le Docker daemon traite les requêtes API afin de gérer les différents aspects de l'installation comme les images, les conteneurs ou les volumes de stockage.

Les images Docker sont des modèles en lecture seule, utilisées pour créer des conteneurs Docker. Elles sont composées de plusieurs couches empaquetant toutes les installations, dépendances, bibliothèques, processus et codes d'application nécessaires.



Le Docker registry

Le registre Docker est un système de catalogage permettant l'hébergement et le "push and pull" des images Docker. Vous pouvez utiliser votre propre registre local ou l'un des nombreux services hébergés par des tiers comme Amazon ECR ou Google Container Registry.





Les avantages de Docker

Docker offre plusieurs avantages par rapport aux méthodes de déploiement traditionnelles. Tout d'abord, il permet de créer des conteneurs légers et portables qui peuvent être déployés rapidement et facilement sur différentes plateformes. De plus, Docker permet d'isoler les applications et leurs dépendances, ce qui facilite la gestion et le déploiement des applications. Enfin, Docker offre une grande flexibilité et facilite le travail en équipe en permettant aux développeurs de travailler sur des applications de manière indépendante et de partager leurs images et leurs conteneurs avec d'autres développeurs.



Travailler avec des conteneurs Docker

Une fois que vous avez une compréhension de base de Docker, vous pouvez commencer à travailler avec des conteneurs Docker. Cette section explique comment créer des images Docker et exécuter des conteneurs.

Attention vous allez avoir besoin d'installer docker sur votre machine pour faire les commandes de cette partie du cours.



Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
WORKDIR /app
COPY requirements.txt
RUN pip install - --no-cache-dir -r requirements.txt
COPY .
CMD ["python", "app.py"]
```

Dockerfile crée une image qui exécute une application web Python



Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - --no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

FROM spécifie l'image de base que vous allez utiliser (celle présente sur DockerHub)

Dockerfile crée une image qui exécute une application web Python



Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - --no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

WORKDIR permet de spécifier le répertoire dans lequel seront basées les instructions qui suivront son appel

Dockerfile crée une image qui exécute une application web Python



Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - -no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

COPY permet d'importer des documents locaux mais pas ceux provenant d'une URL. (contrairement à ADD)

Dockerfile crée une image qui exécute une application web Python



Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - -no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

RUN permet d'exécuter des commandes supplémentaires à l'intérieur du build du dockerfile

Dockerfile crée une image qui exécute une application web Python

Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - --no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

CMD sert à mentionner les instructions qui seront exécutées en premier lors du lancement du conteneur. Permet une exécution sans paramètre supplémentaire

Dockerfile crée une image qui exécute une application web Python



Créer une image Docker

Ce Dockerfile commence par spécifier une image de base, dans ce cas python:3.8-slim-buster. Ensuite, il définit le répertoire de travail pour l'image, copie le fichier requirements.txt et installe les dépendances avec pip. Enfin, il copie le reste du code de l'application et définit la commande pour exécuter l'application.

Une fois que vous avez écrit le Dockerfile, vous pouvez créer l'image avec la commande **docker build**. Par exemple :

```
$ docker build -t myimage:1.0 .
```

Cette commande crée une image avec le tag **myimage:1.0** à partir du Dockerfile situé dans le répertoire courant (.).




Exécuter un conteneur Docker

Une fois que vous avez créé une image Docker, vous pouvez l'exécuter en tant que conteneur avec la commande **docker run**. Par exemple :

```
$ docker run -p 5000:5000 myimage:1.0
```

Cette commande exécute un conteneur à partir de l'image **myimage:1.0** et mappe le port 5000 du conteneur sur le port 5000 de l'hôte.



Configuration de l'environnement d'un conteneur Docker

Vous pouvez configurer l'environnement d'un conteneur Docker en utilisant les variables d'environnement ou les fichiers de configuration. Par exemple, vous pouvez définir une variable d'environnement dans un Dockerfile :

```
ENV MY_VAR=myvalue
```

Vous pouvez également spécifier des variables d'environnement au moment de l'exécution du conteneur avec la commande **docker run**. Par exemple :

```
docker run -e MY_VAR=myvalue myimage:1.0
```

Cette commande exécute un conteneur à partir de l'image **myimage:1.0** et définit la variable d'environnement **MY_VAR** à **myvalue**.



Travailler avec des volumes Docker

Les volumes Docker sont utilisés pour stocker des données persistantes en dehors du conteneur. Par exemple, vous pouvez utiliser un volume Docker pour stocker des données de base de données ou des fichiers de configuration.

Pour créer un volume Docker, vous pouvez utiliser la commande `docker volume create`. Par exemple :

```
docker volume create myvolume
```

Cette commande crée un volume Docker appelé `myvolume`.

Pour monter un volume Docker dans un conteneur, vous pouvez utiliser la commande `docker run` avec l'option `-v`. Par exemple :

```
docker run -v myvolume:/data myimage:1.0
```

Cette commande monte le volume Docker `myvolume` dans le conteneur et le rend accessible dans le répertoire `/data`.



Travailler avec des réseaux Docker

Les réseaux Docker permettent aux conteneurs de communiquer entre eux et avec le monde extérieur. Vous pouvez créer des réseaux Docker pour isoler vos applications et les protéger contre les attaques.

Pour créer un réseau Docker, vous pouvez utiliser la commande **docker network create**. Par exemple :

```
docker network create mynetwork
```

Cette commande crée un réseau Docker appelé **mynetwork**.

Pour exécuter un conteneur Docker sur un réseau spécifique, vous pouvez utiliser l'option **--network** avec la commande **docker run**. Par exemple :

```
docker run --network mynetwork myimage:1.0
```

Cette commande exécute un conteneur à partir de l'image **myimage:1.0** sur le réseau **mynetwork**.



Limites et alternatives à Docker

Docker peut être plus difficile à mettre en place que d'autres technologies de virtualisation, en particulier pour les utilisateurs débutants.

Les conteneurs Docker partagent le même noyau de système d'exploitation, ce qui peut créer des problèmes de sécurité si le noyau est compromis.

Les conteneurs Docker peuvent être moins isolés que les machines virtuelles, ce qui peut causer des problèmes de performance ou de sécurité dans certaines situations.




Playground & TP Docker

Pour la suite et les exercices de ce cours vous devez effectuer le cours docker 101 :

<https://www.docker.com/101-tutorial/>

Puis :

<https://codedesign.fr/docker-desktop-windows-debutant/>



Les outils classiques (Terraform, Ansible, Chef, Puppet)



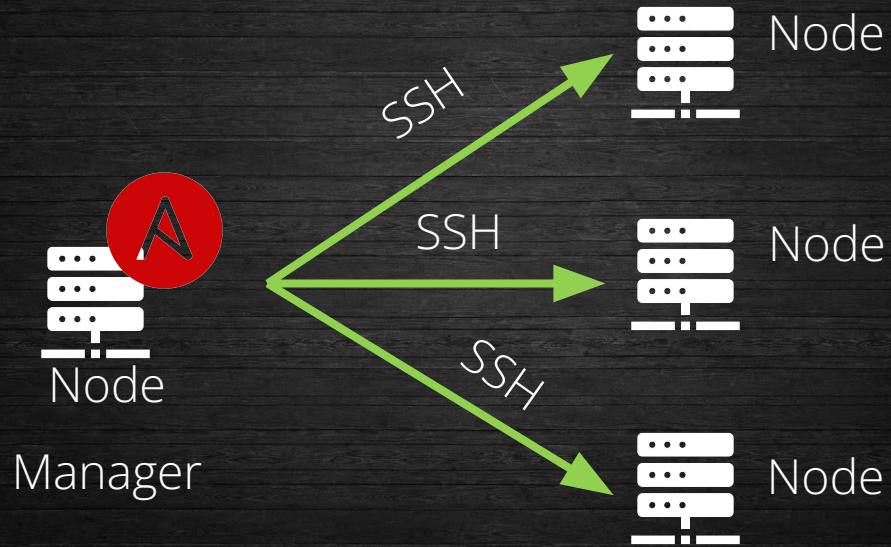
Ansible

Un outil IaC !

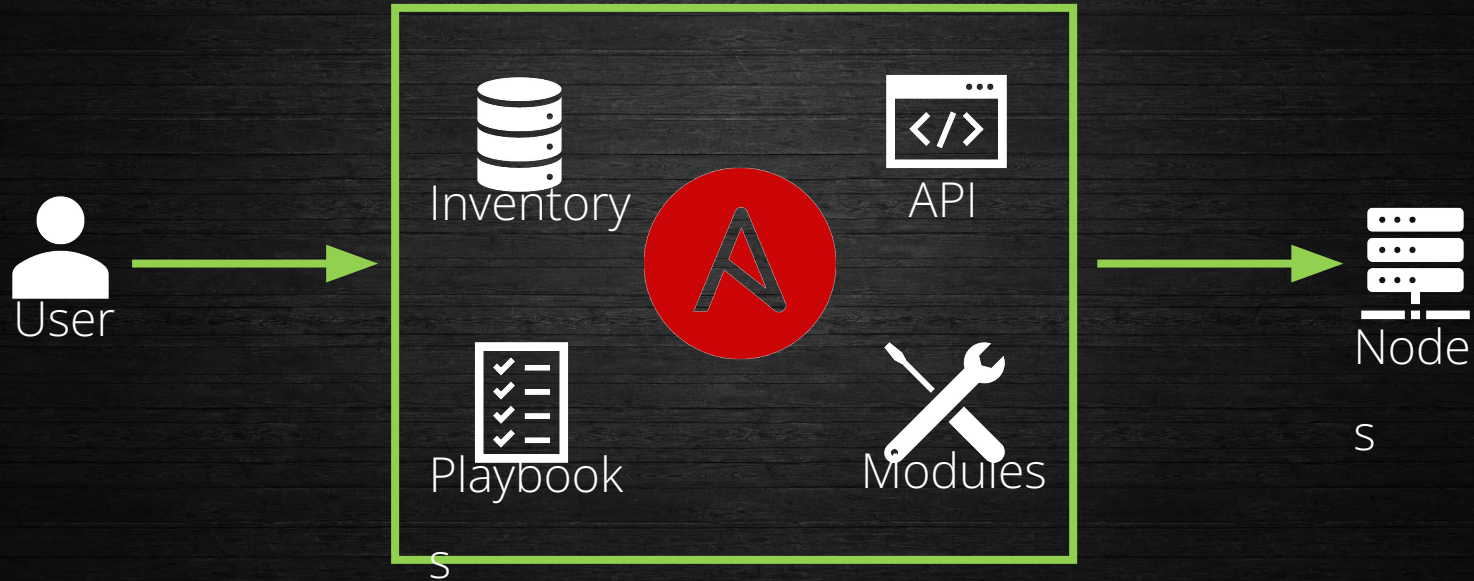
Présentation Ansible



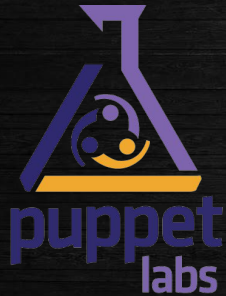
Architecture



Comment ça marche ?



Pourquoi Ansible ?



Adapté pour l'infra Réseau

Nombre de modules

importants

Agentless

Playbooks

```
---  
  
- name: Network Getting Started First Playbook  
  connection: network_cli  
  gather_facts: false  
  hosts: all  
  tasks:  
  
    - name: Get config for VyOS devices  
      vyos_facts:  
        gather_subset: all  
  
    - name: Display the config  
      debug:  
        msg: "The hostname is {{ ansible_net_hostname }} and the OS is {{ ansible_net_version }}"
```

TP Ansible

<https://linux.goffinet.org/ansible/monter-lab-linux-ansible/>

Rendu attendu : Rapport de TP PDF : Deadline : 17 mars 2024 - 23h59